# IEICE TRANSACTIONS

## on Information and Systems

**VOL.E82-D**
**NO.1**
**JANUARY 1999**

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY

The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105 JAPAN

105-01820/



PAPER *Special Issue on New Generation Database Technologies*

# SAMDW–Software Agents Meet Data Warehouses, New Generation Data Warehouse Technologies

Zakaria MAAMAR[†*], *Nonmember*

**SUMMARY** The paper investigates several approaches for designing and implementing integration environments. Such an environment is developed for the purpose to allow cooperative interactions between distributed and heterogeneous systems. A possible approach to achieve system integration is to use the warehousing technology which engenders the development of data warehousing environments. These environments are information repositories that are available for queries and analysis. In order to manage efficiently a data warehouse, software agents enhanced with mobility mechanisms are introduced. A software agent is an autonomous entity having the abilities to collaborate with each other and to answer users' needs. Furthermore, to perform their operations software agents can migrate off their hosts and roam the network to gather relevant information. This research is part of the SAMDW project which aims at developing a new generation of data warehouses.

***key words:*** *data warehouse, software/mobile agent, distribution, heterogeneity*

## 1. Introduction

With the evolution of information and communication technologies, providing users with relevant and up-to-date data becomes a complex task. These data are available on several *information sources* which are *distributed* over networks and *heterogeneous* at different levels (hardware, software, and terminology). It is, therefore, necessary to find ways for *integrating* these systems.

There exists a number of studies in the field of system integration. In fact, two approaches aiming at providing access to distributed and heterogeneous information sources are usually proposed [18]: the *lazy* approach where information are extracted only when queries are asked, and the *data warehousing* approach where the repository is used as a warehouse storing information of interest [6]. However, there doesn't exist a *design approach* that can orient designers when applying the agent technology to the data warehousing field. Therefore, our work aims at providing such an approach. This work is part of the SAMDW (Acronym of Software Agents Meet Data Warehouses) research project that deals with the application of cooperative agents for information management in constrained environments.

Currently, the data warehousing approach is becoming one of the leading issue in the database community and industry. Such an approach is appropriate for [18]:

- Clients requiring specific, predictable** portions of the available information. From our point of view, clients mean mid and upper level managers.
- Clients requiring high query performance, but not necessarily requiring the most recent state of the information. Performance is ensured because the local information sources are not necessary to perform this query.

An integration environment based on a data warehousing approach has to keep local information sources *autonomous* and *independent*. To this end, and as a possible solution, we suggest to introduce several specialized components, called *software agents* [10]. These agents are the front-ends of the information sources and have the capability to act on their behalf. In addition, given the complexity of managing distributed environments, we suggest to enhance some of these agents with *mobility* mechanisms. The mobile agent paradigm is a new trend in Distributed Artificial Intelligence (DAI) research [12]. The basic idea in this paradigm is to allow an agent to move from one host system to another in order to perform specific operations. Each host belongs to a node of a communication network, such as the Internet. Generally, the agent migration is based on the availability of resources, e.g. information sources, that the agent needs to perform its operations. Mobile agents have already been used in a number of areas including [16]: information gathering, data mining, work flow, and electronic commerce.

In this paper, we present the SAMDW project, currently in progress at the Systems Design Engineering Department of University of Waterloo and Research Center in Geomatics of University of Laval. One of the

Manuscript received March 23, 1998.
Manuscript revised August 17, 1998.

[†]The author is with the Systems Design Engineering Department, University of Waterloo, Waterloo, ONT N2L 3G1, Canada, and with the Research Center in Geomatics, Laval University, Ste-Foy, QC G1K 7P4, Canada.
[*]Presently, the author is with the Interoperability Group, Information Section Technology, Defence Research Establishment Valcartier, 2459 Pie-XI Blvd North, Val-Bélair, QC, G3J 1X5, Canada.

**Predictable because the data warehouse is designed according to the clients' needs.

goals of the project is to rely on different techniques borrowed from DAI and applied to the data warehousing technology.

The remainder of this paper is organized as follows. Section 2 proposes guidelines to deal with integration issues, based on a number of techniques such as software agents, ontologies, and mobile agents. Section 3 gives an overview of the warehousing field. Section 4 describes two research perspectives conducted in the context of the SAMDW project. Section 5 discusses the implementation aspects related to these two perspectives. Finally, Sect. 6 summarizes the paper by giving an insight on topics that will be tackled in our further research.

## 2. Integration Issues

The motivation behind the development of an integration environment is to facilitate the exchange of information and services between distributed and heterogeneous systems. However, to develop such an environment, we have to deal with the issues illustrated in Fig. 1 and discussed as follows:

- Satisfying *users' needs* is a complex operation for users and designers. Both of them have to take into account the structural and functional characteristics of each system required in such an operation (Fig. 1 (a)).
- *Knowledge* disparity has to be resolved in order to avoid the misunderstanding situations. Indeed, each system may be part of an organization that has its own needs, specific terminology, and processing procedures (Fig. 1 (b)).
- *Distribution* constraint results from the rapid development of communication means and their reliability and efficiency (Fig. 1 (c)).

In the SAMDW project, we considered the following research domains to overcome the issues presented above.

*Software agents for users' needs issue*—in DAI, researchers have studied a broad range of issues related to the distribution and coordination of knowledge and actions in environments involving multiple *agents* [9].

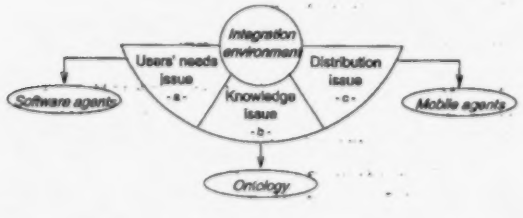


**Fig. 1** Issues related to an integration environment.

These agents, can be thought of as collectively forming a society. Agents can take different forms depending on the nature of the environment in which they evolve [8]. A particular type of agents, *software agent*, has recently attracted much attention [1], [2]. Software agents are autonomous entities having the abilities to assist users when performing their tasks, to collaborate with each other to jointly solve different problems, and to answer users' needs.

*Ontology for knowledge issue*—to interact efficiently, integrated systems should agree on the conventions used for representing and understanding the knowledge they exchange. The formalization of this knowledge is part of the *ontology* domain [7], [15]. In the real world, ontological disparities exist at different levels. First, being generally developed independently, systems may use different terms to describe their data; these different terminologies make it difficult for users to use several systems simultaneously. Moreover, a user has to express his needs according to his own vocabulary and understanding of the application domain. Hence, an ontology will allow to harmonize the terminology used in the representation of the used and exchanged knowledge.

*Mobile agents for distribution issue*—with the huge development of communication technology, such as the Internet, the mobile agent technology has attracted a lot of attention for its flexibility and efficiency [12]. This technology is considered as a new alternative to the traditional client-server operating mode. A mobile agent, for example, can migrate off a host and roam the Internet to gather information for its user. It can efficiently access the needed resources since it moves to their network location rather than transforming multiple requests and responses, sometimes across a low-bandwidth connection. Several researchers believe that mobile agents have the potential to provide a convenient, efficient, and robust programming paradigm for distributed applications [12].

## 3. An Overview of Data Warehouses

*Warehousing* is an emerging technology in the field of data integration from multiple information sources. Indeed, this technology aims at developing an information repository, called a *data warehouse*, that can be available for queries and analysis. Furthermore, a data warehouse is viewed as a set of *materialized views* over the data contained in these sources. Each view satisfies specific users' needs and results from the application of data manipulation operations, e.g. selection, join, on the information sources.

The operating mode of a data warehouse is described as follows [18]:

- Information from each source is extracted in advance, translated and filtered as appropriate,

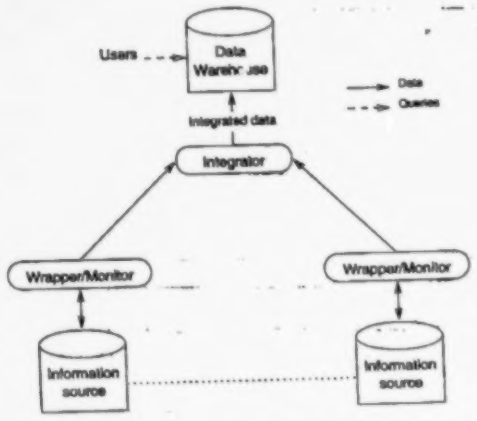


**Fig. 2** Traditional architecture of a data warehouse.

merged with information from other sources, and finally stored in the data warehouse.

- When a query is asked, it is evaluated directly at the repository level, without accessing to the information sources.

Figure 2 presents the basic architecture of a data warehouse. At the lowest level, multiple types of information sources can be available, such as flat files, knowledge bases, database systems, etc. Moreover, three main components exist in the data warehouse: the *Wrapper*, the *Monitor*, and the *Integrator*.

- Wrapper component: it maps information from the native format of the source into the format and data model used by the data warehousing system.
- Monitor component: it detects updates that occur in the information source and notifies them to the Integrator.
- Integrator component: it manages the information in the data warehouse. Hence, this component filters the information, summarizes it, or merges it with information from other sources.

To summarize, a data warehouse is simply a single, complete, and consistent store of data obtained from a variety of distributed and heterogeneous information sources and made available to end users [6].

## 4. Research Perspectives in the SAMDW Project

The SAMDW project is currently focussed on the development of two systems, called respectively MDW for Managing a Data Warehouse and QDW for Querying a Data Warehouse.

### 4.1 MDW System: Managing a Data Warehouse

The main idea in the data warehousing approach is to store relevant integrated information in *advance* for future querying activities. Such an approach avoids accessing the original sources when answering a query; hence, reducing processing time especially if the involved sources are multiple and remote.

Obviously, an *update* detected by the Monitor component in a specific source (Fig. 2) has to be propagated to the data warehouse through the Integrator component. However, in order to apply this update the Integrator may need to get more information from the same or different sources. In this situation, one potential *drawback* of the warehousing approach results from a possibly large volume of information to be transfered several times from the source level to the integration level. As a possible solution, *software agents* enhanced with *mobility mechanisms* and *scenarios techniques* can be used. If an update occurs in an information source the Monitor/Wrapper, *leveraged* to an agent, asks for the scenario† that matches this update. If this scenario mentions that other information sources are needed, the Monitor/Wrapper *migrates* to these sources and *acts as* an Integrator (Fig. 3).

A data warehouse results from the integration of several views, that involve one or several information sources. Furthermore, each view is already known by the data warehouse's administrator; he knows in advance from which sources data should be extracted, and which data should be copied and integrated. As suggested in Fig. 3, software agents are used in the design and management of the MDW system. In this environment, not all the agents need to be mobile. For instance, a static agent is needed to manage the scenarios that describe how to update the data warehouse. Furthermore, this agent interacts with the data warehouse's administrator.

Leveraging a data warehouse's components using the agent paradigm is a research perspective in the warehousing field. Agent and agent mobility technologies allow to reduce the volume of data to be transfered from the information sources o the data warehouse. In fact, all the integration operations are performed at the source level. Previously these operations are done by the Integrator component that may need to fetch data several times from the information sources (Fig. 2). Furthermore, with the development of the hardware technology (computer cost, hard disk capacity, processor speed, etc.), the predefined scenarios of the MDW system can optimize the integration operations by taking into account the characteristics of each information

†Because a data warehouse is a set of materialized views, each view can be described in advance. However, when users have unpredictable needs, using scenarios is not efficient. This is what the project QDW tries to tackle.

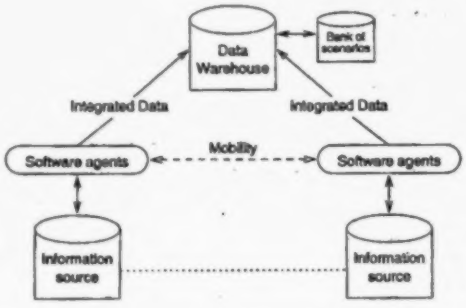


**Fig. 3** Preliminary architecture of the MDW system.

source, in terms of the source nature (database system, flat file, etc.), the implementation language, the data management system, the hardware performance, the network type (LAN vs. WAN), etc. For example, it will be beneficial for efficiency sake, to perform the integration operations taken into account each computer load.

Another aspect in the MDW system is to use a distributed approach for managing the scenarios. In Fig. 3, all the scenarios are in a *bank* that belongs to the data warehouse. However, these scenarios can be delegated to the information sources. Hence, each update in a specific information source is described by its own scenario.

To summarize, two main questions are identified in the MDW system and are under investigation:

- How to use the agent technology in order to design a data warehouse?
- How to use mobile and scenario technologies in order to specify the operating mode of this data warehouse?

In the following sub-sections, we provide answers to these two questions by describing the MDW system in terms of architecture (cf. Sect. 4.1.1) and operating mode (cf. Sect. 4.1.2).

### 4.1.1 MDW's Architecture

For the design of the MDW integration architecture, we have applied the same principles elicited in the field of information systems interoperability [3]. These principles are summarized as follows:

- *Autonomy*: information sources in the integration environment need to have the flexibility to be designed, constructed, and managed independently, without having to conform to this environment characteristics.
- *Adaptability*: information sources that use either standard or non-standard technologies as well as new and legacy sources, can be incorporated into the integration environment in a seamless way without causing any disruption to the current environment.
- *Scalability*: the total integration environment must be expandable and allow an incremental development, such that the integration can start with a number of sources and gradually extend over time, without loosing integrity.

Figure 4 presents the architecture of the MDW system. In this architecture, each information source is encapsulated into a MAS that maintains the autonomy and independence of this source to be integrated in the data warehouse.

According to the principles described previously and to MDW's architectural characteristics, different types of agents are identified:

- *Monitor-Agent*: it monitors changes occurring in the information source when it is updated†. Updates are the result of the instantiation operations performed by users.
- *Scenario-Agent*: it manages the scenarios that specify the operations needed to carry out updates of the data warehouse content. These scenarios are part of the *Scenario-Base* that is accessible through the Scenario-Agent. Furthermore, this agent interacts with the administrator during the update of the Scenario-Base.
- *Wrapper-Agent*: it knows protocols through which an information source accepts requests and provides back information. Furthermore, this agent translates information from the native format of the source into the format and data model used by the data warehouse. According to an update scenario, it occurs that the Wrapper-Agent acts as an *Integrator-Agent* when it migrates to a remote MAS and meets other Wrapper-Agents (cf. Sect. 4.1.2).

In an environment whose agents are mobile, *mobility* operations consist in *transferring* agents through the net to other distant systems, *authenticating* these agents as soon as they arrive, and finally *installing* these agents to resume their activities. To perform these operations in the MDW system, we introduce another type of agents, called *Server-Agent* in the proposed architecture of Fig. 4. For instance when the Wrapper-Agent decides to move, according to the proposed scenario, it interacts first with the Server-Agent in order to be transfered to the desired system.

†Hence, a Monitor-Agent knows protocols managed this information source.

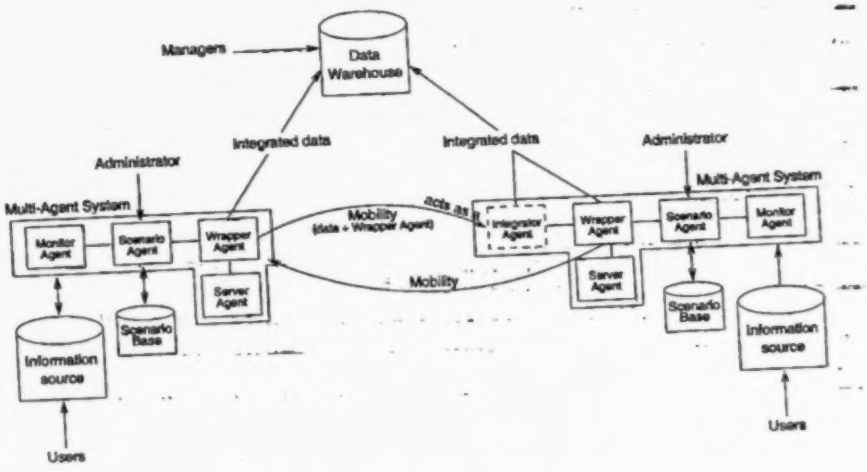


**Fig. 4** Architecture of the MDW system.

### 4.1.2 MDW's Operating Mode

In Fig. 4, agents interact in several ways. Indeed, the Monitor-Agent notifies the Scenario-Agent with the update that it has detected in its associated information source. Next, the Scenario-Agent browses its Scenario Base (it is specific to each information source) and sends to the Wrapper-Agent the scenario that corresponds to the update mentioned by the Monitor-Agent. Using the proposed scenario, the Wrapper-Agent accesses the source and sends the updated information to the data warehouse. Such a situation corresponds to the case where the proposed scenario involves only the updated information source.

More interesting are the situations requiring the integration of information from several distributed and heterogeneous sources. According to the scenarios applying to these situations, the Wrapper-Agent migrates to other MASs and acts as an Integrator-Agent. When this Integrator-Agent arrives, it requests information from the Wrapper-Agent of the current MAS (the Wrapper-Agent uses its associated source). It also happen that in order to have an optimized scenario, the Wrapper-Agent invites another Wrapper-Agent to migrate to its own MAS. This situation is called an *invitation*.

The operating mode of the MDW system is specified using several *scenarios* that depend on the nature of the information source's updates. We model a scenario as a *network* whose *nodes* correspond to the couple (MAS, Information-Source) and whose links correspond to the migration operations between the nodes. For example, Fig. 5 illustrates a scenario that involves three information sources (IS-1, IS-2, IS-3) and their MAS (MAS-1, MAS-2, MAS-3). Numbers in this figure indicate the chronology order of operations processing. Once the Monitor-Agent of MAS-1 has detected (0) an update in the IS-1 (Fig. 5), it uses the appropriate scenario provided by the Scenario-Agent. This scenario mentions that the Wrapper-Agent of MAS-1 has to move to MAS-2. When this agent arrives at MAS-2 (1), it acts as an Integrator-Agent and collaborates with the Wrapper-Agent of MAS-2. Each agent, Integrator and Wrapper, provides its own data in order to perform the integration operation (2). Furthermore, the proposed scenario mentions that other data from IS-3 are needed before the update of the data warehouse can be achieved. Hence, the Integrator-Agent, which is currently in MAS-2, invites (3) the Wrapper-Agent of MAS-3 to move (4) with relevant data from IS-3. The partial results provided in step 2 are integrated with the data of IS-3 (5). Finally, the data warehouse is updated (6).

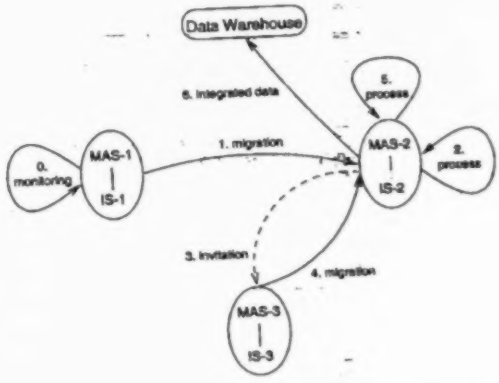


**Fig. 5** Example of the MDW system operating mode.

**Specification of The Scenario-Language**

In Table 1, we describe the language proposed for the specification of the MDW system's scenarios. We use a predicate-based representation and the following sym-

bol: $*[\prec\cdots\succ]*$ means $0\cdots n$. In the following paragraph, numbers between parenthesis refer to numbers in Table 1.

A scenario (1) is decomposed into two parts: the head and the body. The first part identifies the information source and the agent related to this scenario (2). The second part is a rule to be validated in order to trigger this scenario (3). If the condition part of this rule is true, then an update (i.e., either an insertion, a modification, or a deletion) occurred in this information source's data (4) and specific operations have to be done (5). These operations consist in processing the updated data, migrating or not to other information sources (6, 7, 8), and finally updating the data warehouse.

#### 4.1.3 Summary

We presented the main characteristics of the MDW system. This latter aims at using scenarios in the specification of a data warehouse operating mode and software agents enhanced with mobility mechanisms in the completion of these scenarios.

### 4.2 QDW System: Querying a Data Warehouse

In Sect. 1, two approaches aiming at providing access to distributed and heterogeneous systems have been presented: the *lazy* approach and the *data warehousing* approach. The lazy approach seems to be less *efficient* than the warehousing approach, because it directly depends on the state of networks. If a "strategic" portion of a network is down, there will be no way to access information sources. To avoid such situations, the data warehouse seems to be an alternative approach. The main idea is to *cache* relevant data for further uses. However, even if a data warehouse provides an integrated information from distributed and heterogeneous systems, it happens that this data warehouse is not *able to answer* all users' queries. The reason is: when a data warehouse is designed, its designers only take into account users' needs that are already *known*. Hence, queries that are not satisfied by a data warehouse represent users' *unknown* needs. Therefore, the QDW System aims at proposing an approach for *dynamically* querying a data warehouse. This work is promising and relatively unexplored, as explained in [18], [19].

**Table 1** Specification language of the MDW system scenario.

```
1 ≺scenario≻   ::= ≺head≻ ≺body≻.
2 ≺head≻       ::= ≺scenario_id≻ ≺source_id≻ ≺agent_id≻
3 ≺body≻       ::= If ≺condition≻ Then ≺operations≻
4 ≺condition≻  ::= insertion | modification | deletion ≺data≻
5 ≺operations≻ ::= process ≺data≻
                   *[∧ ≺operation≻]*
                   update ≺data_warehouse_id≻
6 ≺operation≻  ::= ≺migration≻ | ≺invitation≻
7 ≺migration≻  ::= move ≺agent_id≻ ≺data≻ to ≺source_id≻
                   process ≺data≻
8 ≺invitation≻ ::= invite ≺agent_id≻ ≺data≻ to ≺source_id≻
                   ≺migration≻
```

A data warehouse is an integration environment that provides users with appropriate data from multiple distributed and heterogeneous information sources. To satisfy users' needs, several *pre-defined* queries are proposed by the warehousing environment and are triggered when needed. However, certain users may query the data warehouse's informational content without using the pre-defined queries. Hence, they formulate *ad-hoc* queries. The issue is to verify whether such queries can be satisfied by the data warehouse or not. If the warehouse cannot satisfy them, then the information sources integrated in this warehouse have to be involved. The QDW system focuses on such a situation.

In the following sub-sections, we describe the QDW system in terms of architecture (cf. Sect. 4.2.1), operating mode (cf. Sect. 4.2.2), and finally ontology (Sect. 4.2.3).

#### 4.2.1 QDW's Architecture

The QDW system proposed for querying a data warehouse uses three main concepts: *software agents*, *mobility/remote-processing* techniques, and *ontologies*. This approach is illustrated in Fig. 6 and consists of two sides: the *Data-Providers* and the *Data-Consumers*. Data-Providers side[†] gathers the data warehouse and the information sources, whereas Data-Consumers side identifies users. These two sides are related by mobility or remote processing techniques. Mobility means that the Resolution-Agent, of the Data-Consumers side migrates to the Data-Providers side, interacts with Knowledge-Agents, belonging to this side to process its queries, and goes back with the results it obtained to the Data-Consumers side. Remote processing means that an agent of the Data-Consumers side sends its queries to the agents of the Data-Providers side to process them and send the results they obtained to this agent.

In Fig. 6, several types of agents exist and some of them are enhanced with mobility capabilities. In what follows, we describe each agent's functionalities.

- *Interface-Agent*: it assists users in formulating their needs, forwards these needs to the Resolution-Agent to be satisfied, and provides these users with the answers obtained from the Resolution-Agent. To facilitate the formulation operation, the Interface-Agent provides users with appropriate knowledge of the Ontology-Base.

[†]In the rest of this article, the data warehouse is viewed as an information source.

- *Resolution-Agent*: it processes users' queries. Therefore, the resolution process may require from this agent either to remotely interact with the Knowledge-Agents of the Data-Providers side or to migrate to this side and meet these Knowledge-Agents. A decision about a mobility or a remote processing is based on the number of the information sources required to satisfy users' queries. The Resolution-Agent substitutes the Integrator component presented in Fig. 2.

- *Domain-Agent*: it identifies the appropriate information sources for satisfying users' queries and specifies how the data of these sources are going to be integrated. Furthermore, this agent resolves knowledge disparities that may exist between the several sources using the Ontology-Base. Such an ontology describes the structural and functional characteristics of the available systems, including the data warehouse.

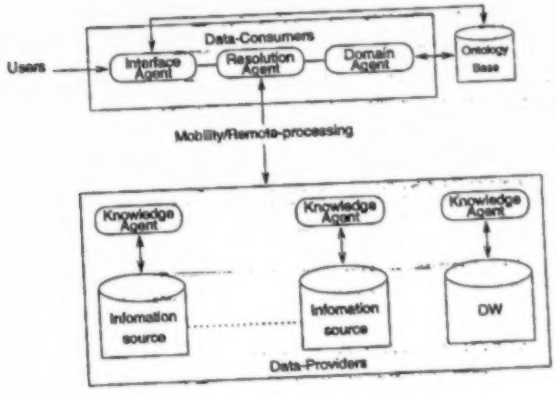


**Fig. 6** Architecture of the QDW system.

- *Knowledge-Agent*: it maintains the autonomy and independence of the information sources and knows the protocols through which a source accepts queries and provides back information. This agent remotely or locally receives users' queries from the Resolution-Agent. For the local case, it means that the Resolution-Agent has migrated to the Data-Providers part. The Knowledge-Agent substitutes the Wrapper/Monitor component presented in Fig. 2.

#### 4.2.2 QDW's Operating Mode

In Fig. 7, the operating mode of the QDW system is summarized by two situations. They are dedicated to ad-hoc user queries' processing and are as follows:

- Only one information source is needed: remote processing technique is used (Fig. 7, Situation a).
- Several information sources are needed: mobility technique is used (Fig. 7, Situation b).

The following paragraphs describe the operations (solid and dashed lines in Fig. 7) of the QDW system. Numbers in parenthesis correspond to numbers in Fig. 7.

When a user wants to satisfy his needs (0), he invokes the Interface-Agent that connects to a specific part of the Ontology-Base. This connection is useful; it helps the user to specify his needs and to overcome the knowledge heterogeneity constraint. Next, a user's needs are mapped to a query transmitted to the Resolution-Agent (1). This agent asks for details on the sources required to satisfy this query. To achieve this operation, the Resolution-Agent interacts with the Domain-Agent (2) that accesses its Ontology-Base. Once the information sources to be requested

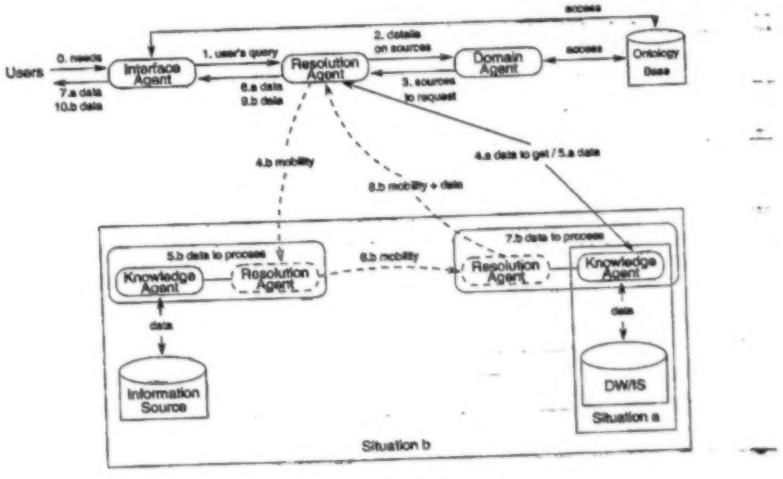


**Fig. 7** Operating mode of the QDW system.

have been identified by the Domain-Agent (3), two situations exist and are identified in Fig. 7 with letter a and letter b.

In Situation a, the Resolution-Agent sends remotely (4.a) the user's query to the Knowledge-Agent of the source suggested by the Domain-Agent. The Knowledge-Agent accesses this source and provides the data it obtained to the Resolution-Agent (5.a). Next, these data are sent to the user through the Interface-Agent (6.a, 7.a).

In Situation b, a set of multiple information sources is needed to satisfy the user's query. The data warehouse can be part of this set. First, the Resolution-Agent moves to the first source of the proposed set (4.b) and interacts with the Knowledge-Agent associated to this source to get appropriate data (5.b). Next, the Resolution-Agent moves again to another source (6.b) until it has visited all the sources of the set proposed by the Domain-Agent (7.b). At the end of the Resolution-Agent's *trip*, it goes back to its initial system, i.e., the Data-Consumers side, and provides the user with the results it obtained through the Interface-Agent (9.b, 10.b).

The rest of this section is dedicated to Situation b of Fig. 7. In such a situation, the issue is how to integrate the data obtained from distributed and heterogeneous information sources, including the data warehouse. Mobility aspects are also a main aspect of this issue.

Generally, in an integration environment a user's needs are identified by *a global query* that is decomposed into a set of *sub-queries*[4]. Such decomposition depends on the number of the required information sources. Furthermore, the data obtained from the performance of these sub-queries are integrated at the user level. These information sources do not possess appropriate knowledge to process their own data and are not able to cooperate between each other. To overcome these drawbacks, mobile agents are appropriate. Such an agent will have to migrate from a source to another, execute each sub-query, get data, process them (for example, a join operation), and carry them to the next host. The chronology of this agent's operations is illustrated in Fig. 8. In this figure, the abbreviation (y/n) means yes or no, i.e.; when an agent moves, it is not obliged to perform join operations at the new host. It may visit that host to only get appropriate data and then migrates again.

Sub-queries resulted from the global-query decomposition are part of the components that constitute the mobile agent's *itinerary*. These components are related to the information sources to visit, the data to get according to these sub-queries' needs, and the link operations that ensure the integration of the data obtained from heterogeneous sources. These links have already been described in the Ontology-Base.

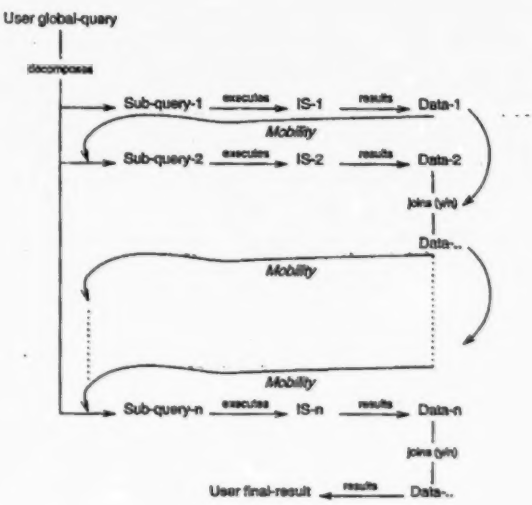


**Fig. 8** Decomposition and performance of a user global-query.

Itinerary Specification

$Itinerary = < IS, SQ, Link >$

- $IS$: set of information sources to visit that has an order;
  $IS = \{IS_1, \ldots, IS_i, \ldots, IS_n\}$
  $IS_{n-1_{(n>1)}} < IS_n$: an agent visits $IS_{n-1}$ before $IS_n$
  $\exists\, i,\ IS_i = DW$

- $SQ$: set of sub-queries to perform;
  $SQ = \{SQ_{IS_1}, \ldots, SQ_{IS_i}, \ldots, SQ_{IS_n}\}$

- $Link$: set of information sources in which join operations are performed;
  $Link = \{IS_i, \ldots, IS_n \mid i > 1\}$; $Link \subseteq IS$
  $Link$ is identified by the word "joins" in Fig. 8.

- Algorithm

```
Let
Pos(i)    :  a function that return the element i of the set IS;
host      :  a variable;
move-to   :  mobility function;
perform   :  performance function;
join      :  join function;
for i = 1 to ||IS||
do begin
    host = Pos(i);
    move-to(host);
    perform(SQ_host);
    if host ∈ Link
    then join;
end
```

#### 4.2.3 QDW's Ontology Aspect

One of the main characteristics of an integration environment is the knowledge disparity that may exist between the information sources of this environment. Indeed, each source belongs to a specific organizational structure that has its own functioning standards. The aim of an integration environment is to reduce this disparity and hence, to reduce the complexity of the operation that involves multiple, distributed, and heterogeneous sources. In a data warehouse, the Wrapper-Component associated to each information source handles the issue of knowledge disparity and provides a coherent knowledge to the Integrator-Component. Hence, an ontology, viewed as an explicit structure, doesn't exist in a data warehousing environment. The ontology is only a set of *mapping rules*.

In Fig. 7, the Ontology-Base is used in order to identify the information sources, including the data warehouse, needed to satisfy users' ad-hoc queries. Interesting are the situations requiring the combination either of a data warehouse with other sources or multiple sources (excluding the data warehouse). In such situations, using the Ontology-Base becomes mandatory; it specifies how the several sources are semantically linked and how the integration operations are going to be performed in real time.

Figure 9 presents the main components of the Ontology-base and its role in the QDW system. Currently, a data warehouse can be considered as a "weak" ontology. A warehousing environment resolves the knowledge disparity at the format level using mapping rules. However, it doesn't provide many details on this knowledge in terms of meaning, implementation, etc. To overcome such constraints, a data warehouse can be substituted by an Ontology-Base enhanced with other types of knowledge described as follows:

- *Conceptual structure*: this knowledge makes an inventory of the components used in the conceptual schema of each source, such as the entity names, the association names, the attribute names, the association types, etc.

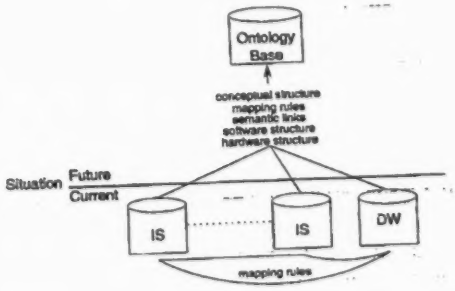


Fig. 9 Ontology-base in the QDW system.

- *Mapping rules*: this knowledge explains the process used to map an information from one format to another format, for example from the inch to the centimeter. These mapping rules integrate also the rules already existing in the data warehouse.

- *Semantic links*: this knowledge lists the structures that are different at the format level, but are the same at the meaning level.

- *Software/Hardware structure*: this knowledge details the tools used in the implementation of each source, for instance the data base management system type, the programming language type, etc.

#### 4.2.4 Summary

We presented the main characteristics of the QDW system. This latter aims at dynamically querying an integration environment based on a data warehouse. To perform this operation, software agents and remote processing/mobility techniques are used.

## 5. Implementation Issues in the SAMDW Project

Several issues related to the implementation of the MDW and QDW systems have to be addressed:

- How to implement MDW's and QDW's agents?
- How to allow MDW's and QDW's agents to move?
- How to allow MDW's and QDW's agents to access information sources?
- And, how to take into account the concurrecy control at the scenario level?

Currently, several programming languages for software/mobile agents exist, such as Java, Tcl, Telescript, and so on. We plan to base our development on the Java programming language[†]. However, in spite of its position as an Internet programming language leader and *portability* advantage, Java is perhaps the "weakest mobile" language [5]; agents in Java (which are viewed as applets) travel from their home to other hosts, and then cannot travel further [13]. To handle this Java's constraint and allow agents to move several times, we plan to use the Object Request Broker (ORB)[††] Voyager from the ObjectSpace Company [17]. Voyager claims to be the world's first 100% Java agent-enhanced ORB;

[†]Java positive aspects are numerous: it is object-oriented; it is safe and secure; it is well supported by major companies; etc.

[††]An ORB is a software that respects the conventions defined by the Object Management Group [11] for the development of an object-oriented distributed environment.

it allows Java programmers to quickly and easily create sophisticated network applications using both traditional and agent-enhanced distributed programming techniques. With Voyager, agents can move to perform high-speed local communications. Furthermore, with Voyager an agent moves by using the *move()* message, supplied with the address of the destination host and the call-back function as parameters. The agent waits until all pending operations are processed, then moves to the specified destination leaving behind a *secretary* to forward messages and future connection requests.

The MDW's and QDW's agents, particularly Monitor-Agent, Wrapper-Agent, and Knowledge-Agent, have to access information sources. Generally, in order to access a source, for example a database system, the mostly used protocol is Microsoft's Open Database Connectivity (ODBC) driver. However, as a result of the rapid development of Internet and Intranet technologies, another protocol has been suggested: the Java Database Connectivity (JDBC) driver [14]. JDBC is a Java application programming interface designed for database access. It allows Java programmers to access data from remote or local databases through one common platform-independent interface.

## 6. Conclusion

In this paper we presented the major characteristics of the SAMDW project that aims at applying software agents enhanced with mobility mechanisms to warehousing environments. We particularly focussed our description on two systems, MDW and QDW, that are currently under investigations. The results provided are part of a larger project whose objective is to use cooperative agents in constrained environment. Data warehousing systems have become a key component in the enterprise strategy for information technologies.

The SAMDW project is still an on going research. We are currently implementing the MDW and QDW prototypes. These prototypes use JAVA as a programming language, the JDBC driver to connect the available informational sources, and finally the ORB Voyager to specify the behavior of the MDW's and QDW's agents during distributed and mobility operations.

### References

[1] Special issue on intelligent services, Commun. ACM, vol.37, no.7, 1994.
[2] Special issue on intelligent internet services, IEEE Expert, vol.10, no.4, 1995.
[3] G. Babin, C. Hsu, and Z. Maamar, "A distributed metadatabase architecture for an enterprise scalable, adaptive integration environment," Technical Report 37-94-424, DSES, Rensselaer Polytechnic Institute, Troy, N.Y., USA, 1994.
[4] W. Cheung and C. Hsu, "The model-assisted global query system for multiple databases in distributed entreprises," ACM Trans. Informations Systems, vol.14, no.4, pp.421–470, Oct. 1996.
[5] G. Cugola, C. Ghezzi, G. Picco, and G. Vigna, "A characterization of mobility and state distribution in mobile code language," Proc. 2nd ECOOP Workshop on Mobile Object Systems, pp.10–19, Linz, Austria, July 8–9, 1996.
[6] B. Devlin, "Data Warehouse from Architecture to Implementation," Addison-Wesley, 1997.
[7] T.R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Technical Report, Knowledge Systems Laboratory, KSL 93–04, Stanford University, Stanford, California 94305, Aug. 23, 1993.
[8] P. Maes, "Artificial life meets entertainment: Lifelike autonomous agents," Commun. ACM, vol.38, no.11, 1995.
[9] B. Moulin and B. Chaib-draa, "An overview of distributed artificial intelligence," in Foundations of Distributed Artificial Intelligence, pp.3–55, eds.G.M.P. O'Hare and N.R. Jennings Wiley, 1996.
[10] H.-S. Nwana, "Software agents: An overview," Knowledge Engineering Review, vol.11, no.3, pp.1–40, Sept. 1996.
[11] OMG, Object Management Group, http://www.omg.org.
[12] Dartmouth Workshop on Transportable Agents '97 (DWTA '97). Dartmouth College, Hanover, NH, USA, 1997.
[13] J. Ordille, "When agents roam, who can you trust?" Proc. First Conf. On Emerging Technologies and Applications in Communications (etaCOM), May 1996.
[14] P. Patel and K. Moss, "Java Database Programming with JDBC," Coriolos Group Books, 1996.
[15] H. Takeda, K. Ino, and T. Nishida, "Agent communication of multiple ontologies," Proc. FGCS '94 Workshop on Heterogeneous Cooperative Knowledge-Bases, pp.11–124, Tokyo, Dec. 1994.
[16] B. Venners, "Solve real problems with aglets, a type of mobile agent," Technical Report, Netscape World Magazine, May 1997.
[17] Voyager, ObjectSpace, http://www.objectspace.com.
[18] J. Widom, "Research problems in data warehousing," Proc. 4th International Conference on Information and Knowledge Management, (CIKM'95), Nov. 1995.
[19] G. Zhou, R. Hull, R. King, and J.-C. Franchitti, "Data integration and warehousing using h20," IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing, vol.18, no.2, pp.29–40, June 1995.

**Zakaria Maamar** is a Defence Scientist with the Defence Research Establishment Valcartier. He has been involved in several research projects concerning software agents and object-oriented frameworks for interoperability issues. His current research interests are mobile agents and data warehouses. He received his M.Sc. and Ph.D. in computer sciences from the University of Laval, Quebec. zakaria.maamar@drev.dnd.ca

#510608